Processor Interface Protocols for Facilitating Detection-Level Data Fusion

November 1998

Gilbert L. Crouse, Jr.
BBN Technologies
1300 North 17th Street
Arlington, VA 22209

LT John Park
Space and Naval Warfare
Systems Command
4301 Pacific Highway
San Diego, CA 92110-3127

ABSTRACT

Sharing track-level contact information across multiple processors and platforms is relatively commonplace now in combat systems. This has been greatly facilitated by the development of communications standards for passing track-level data. However, for more effective data fusion and for operator confirmation purposes, it is often very desirable to share lower level detection and classification data products as well as portions of raw or semi-processed sensor data. Sharing these types of data is often impeded by the need to translate between the different data formats used by the systems involved. Moreover, merely obtaining connectivity between individual stovepiped processing systems is often nontrivial. The work described in this paper is an on-going attempt to develop standard protocols and data formats for communication of lower-level data products between multiple processing systems. The initial effort for this work has been focused on the Under-Sea Warfare community, but the protocols are not exclusively applicable to sonar systems.

1. INTRODUCTION

Detection, classification, and tracking of modern nuclear and diesel-electric submarines have grown increasingly difficult with improvements in quieting technology. Maintaining reasonable detection ranges has required improvements in sensor systems, processing systems, and also increases in the sheer numbers and types of sensors employed. Obtaining the maximum benefit of greater numbers and types of sensors requires that the operation of these sensors be done in concert. In many current systems, however, there is little communication between different platforms prosecuting the same target or even between different sensor systems on the same platform. Each sensor system has dedicated operators whose only knowledge of the scene is the information provided by their sensor. Data exchange between on-board and off-board systems is

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
01-11-1998	Conference Proceedings	xx-xx-1998 to xx-xx-1998

4. TITLE AND SUBTITLE
Processor Interface Protocols for Facilitating Detection-Level Data Fusion
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Crouse, Jr., Gilbert L. ;
Park, John ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
BBN Technologies
1300 North 17th Street
Arlington, VA22209

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
Director, CECOM RDEC
Night Vision and Electronic Sensors Directorate, Security Team
10221 Burbeck Road
Ft. Belvoir, VA22060-5806

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES
See Also ADM201041, 1998 IRIS Proceedings on CD-ROM.

14. ABSTRACT
Sharing track-level contact information across multiple processors and platforms is relatively commonplace now in combat systems. This has been greatly facilitated by the development of communications standards for passing tracklevel data. However, for more effective data fusion and for operator confirmation purposes, it is often very desirable to share lower level detection and classification data products as well as portions of raw or semi-processed sensor data. Sharing these types of data is often impeded by the need to translate between the different data formats used by the systems involved. Moreover, merely obtaining connectivity between individual stovepiped processing systems is often nontrivial. The work described in this paper is an on-going attempt to develop standard protocols and data formats for communication of lower-level data products between multiple processing systems. The initial effort for this work has been focused on the Under- Sea Warfare community, but the protocols are not exclusively applicable to sonar systems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	7	Fenster, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

generally limited to the exchange of contact position reports. The exchange of lower-level data products holds the promise of improved detection and tracking performance of the aggregate system. For example, individual sensor systems may detect a target well before they can develop a confirmed track. By sharing data at a lower level, the detection information could be used to cue operators of other sensor systems. The capabilities of the group of sensor could then be used to confirm or reject the possible contact and develop a track before the individual sensor system could have done it working in isolation. This holds true for both human operators and computer data fusion systems.

In order to support data fusion efforts as well as other applications, the processor interface protocols have been designed with several data transfer modes including continuous data stream mode, data "snippet" mode, and text or data file transfer mode. In continuous data stream mode, data regularly or irregularly sampled in time is continuously transferred from the originating system to a single or to multiple client systems that receive the data. This is envisioned to support transfer of raw or partially processed data to other systems for display or further processing in near real time. The data snippet mode would be used for recall of portions of data for display or reprocessing. For example, this mode could be used to support an operator recalling sections of data in order to evaluate autodetector or autoclassifier decisions. The file transfer mode is meant to support transfer of arbitrary data between systems. For example, this could include operator notes, image files, or processing parameter files. In addition, an "instantaneous" messaging capability is envisioned to allow operators to send alert messages to other operators in a timely fashion.

The Internet community has defined a wide variety of standard, commercial data transfer protocols. With the military's move toward commercial computing and communications equipment firmly in place, the processor interface protocols working group has attempted to use commercial standards wherever appropriate. The underlying assumption of the working group was that physical communications between processors would be based on some type of IP network.

2. Design Considerations

In considering the desired functionality of the interface protocols, several requirements were identified. These include

1. Continuous transfer of raw or processed data streams between multiple processing systems
2. Transfer of portions or raw or processed data streams between processing systems
3. Transfer of arbitrary files between processing systems
4. Data spooling for snippet recall and data archival
5. Query of available data streams for continuous or snippet transfers
6. Query of parameters associated with particular data streams
7. Notification when parameters associated with a particular data stream change
8. Dynamic hookup of data streams
9. Browsing of participating systems (e.g., "network neighborhood")
10. Immediate transfer of alert messages between processing systems

In addition to the functionality requirements, several constraints were also identified, primarily related to the desire to interface to legacy systems and the limitations of the communications channels.

1. Support for legacy systems
   - Desire minimal intrusion into the processing systems
   - Supporting the interface protocols should not place significant burdens on the resources of the processing systems
   - Must support various processing scheduling approaches used by different processing systems (e.g., data flow, event loop, polling)
2. Support for inter-platform connectivity (i.e., ship-to-ship)
   - Available communications bandwidth will be limited
   - Network connections may be unreliable due to RF communication links

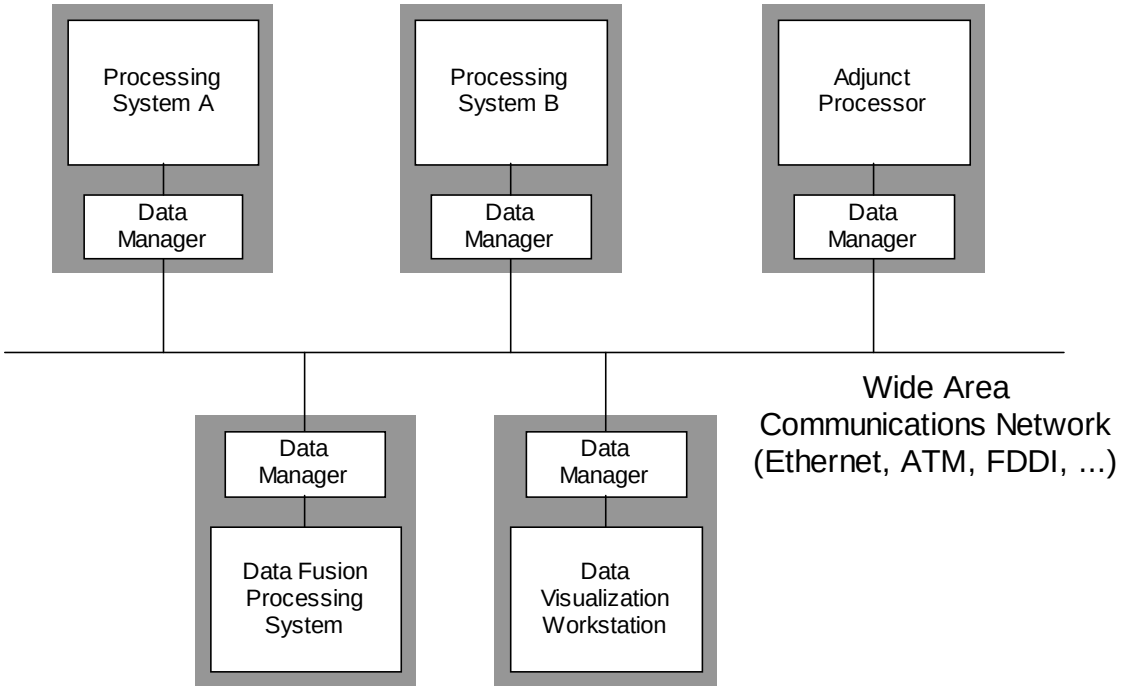

Figure 1. Example cluster of processing systems.

3. APPROACH

3.1. Data Manager

A cluster of processing systems communicating via the processor interface protocol is depicted in Figure 1. Each processing system participating in the cluster is paired with a "data manager" that has several functions, including:

1. File Server. The data manager serves as an HTTP server. Any files that the processing system wishes to make available are handed over to the data manager and the data manager services request for those files.

2. Data Spooler. Any streams of data that the processing system will make available to other systems are transferred in real time to the data manager. The data manager retains a spool of the data on disk for some length of time.
3. Streaming Data Server. Using the RTP/RTSP protocols, the data manager makes streams of data or portions of streams available to the other processing systems. The data manager can continuously transfer a data stream or the client can request that only a portion be transferred. The list of available data streams is maintained by the data manager as a file that can be requested via the HTTP server.
4. Cluster Browser Server. Each data manager maintains a list of all other data managers participating in the cluster. This list is made available on demand to the processing system via HTTP. The list of participating processing systems is maintained through the use of multicast messages. Each participating data manager sends out a multicast message periodically. Each data manager also listens for those messages from other processors in order to determine what other systems are available.
5. Synchronous Messaging Server. The data manager can transmit alert messages to other processing systems. The data manager maintains a list of the systems that are registered to receive the messages and when instructed will send a message to the designated recipient. The message will be transferred immediately and thus will not suffer the indeterminate delays associated with email.

The data manager itself consists of several software programs. Unless they are designed to support it, most processing systems will not have the spare processing and storage resources available to host the data manager. Thus in most cases, the data manager will need to reside on a separate computer or processing board. The data manager is being designed to be portable and to run on UNIX systems or inexpensive Windows NT PCs. This minimizes the additional cost associated with adapting a system so that it can use the processor interface protocols. While it is assumed that the data managers are connected together via a WAN with limited bandwidth, it is assumed that the link between the data manager and its associated processing system are local to one another and have a fairly high-speed link.

The interface between the processing system and the data manager is specified in terms of an application-programming interface. A reference implementation of the interface using an IP socket-based protocol is currently being developed along with the data manager. This interface library supports control of the data manager, transfer of data streams to the data manager, publishing of arbitrary data files (text files, images, parameter files, etc…) through the data manager, and transmission of alert messages to other processing systems.

Since HTTP is the primary protocol used for transfer of data files and control information between systems, standard WWW browsers can be used to view many of the files on the data manager. Each data manager maintains several pages with predefined names that clients can request. These include

1. `/data_managers.html`: This file contains a list of all of the active data managers in the cluster. This file is primarily used by data clients local to the data manager so that they can locate sources of data.

2. `/data_clients.html`: This file contains a list of all of the active data clients in the cluster. This file is primarily used to identify data clients for sending alert messages to other operators.
3. `/data_files/all.html`: This file contains an index of all of the data files that are available on the data manager.
4. `/data_streams/all.html`: This file contains an index of all of the data streams that are available on the data manager.
5. `/data_streams/streamid=???`: This file contains a description of the data stream referenced by the given streamid.

These files use standard HTML formatting and contain hyperlinks to facilitate browsing using a standard WWW browser. In addition, browser add-ins are planned to enable viewing of data files from within a WWW browser.

3.2. Data Clients

Systems that wish to access data from other processor use a data client library that communicates with the other processor's data manager. This client library handles the details of the HTTP or RTSP/RTP protocols. The client library must perform a number of tasks to accomplish data transfers of a continuous data stream.

1. Download list of other data managers from local data manager, parse list and present to processing system or operator.
2. Select appropriate data manager and download list of available data streams.
3. Select appropriate data stream and download description of data format and proper port for delivery.
4. Notify data source of interest in selected data stream.
5. Hookup to data stream.
6. Listen for any changes in data stream parameters.

The client library attempts to make as much of this as possible transparent to the processor application. The library support several different methods of notifying the application about the availability of new data. This is intended to support different scheduling methods used by different application.

3.3. Data Formats

Different processing systems inevitably utilize different formats to represent their data. This presents a problem for transferring data between two systems. The transfer protocols described here are independent of the underlying format of the data. Any data format can be supported. During data transfers, the data format is specified as one of the initial parameters sent between systems. However, in order to facilitate interoperability, a standard format is desirable. One such format that has gained acceptance recently is the Common Acoustic Sensor Data Exchange (CASDE) format. This format has been designed specifically for acoustic sensor array data, but is sufficiently general to handle many types of raw or processed data. This format will be one of

the standard formats used for both data files and also continuous data streams. The CASDE format specifies a header that describes the data in the file. For normal data files, this header is contained at the top of the file. For data streams, this header is stored separately. Clients can request the header to obtain information on the data contained in the data stream, and then hookup to the data stream itself using the RTSP commands. Storing the header separately enables clients to hookup to data streams "on the fly."

4. SUMMARY

Sharing data between different types of sensor systems and sensor systems on different platforms holds the promise of providing significant improvements in detection and tracking performance. The processor interface protocols described in this paper are an attempt to facilitate this data sharing using Internet standard data transfer protocols. The protocols have been developed with an eye toward the under-sea warfare community, but are not restricted to that application and should be applicable to a wide variety of processing systems.

The results of this project will be a specification document describing the processor interface protocols and how they should be used. In addition, a portable data manager and an interface library that serve as reference implementations of the specification will be developed. An effort has been made to ensure that appropriate Internet standards are used for data transfer rather than creating new standards. Recent work on multimedia broadcasting via the Internet has provided a capable infrastructure for transmission of continuous data streams and portions of data streams. In addition, the HTTP protocol has been used wherever possible to support transfer of arbitrary data files between systems. Use of HTTP enables standard WWW browser tools to view the data files available on any of the data managers and to navigate between the data managers in the cluster.

ACKNOWLEDGMENTS

The protocols described in this paper have been designed and developed by the Processor Interface Protocols working group sponsored by the Full Spectrum Program of SPAWAR PD18. The authors wish to acknowledge the contributions of the other working group members including Lisa Blodgett (JHU/APL), Greg Gerding (TRW), gene Hardekopf (TRW), Hilary Hershey (JHU/APL), Jim Ionata (NUWC), James Lockwood (SSC), Jerry Moons (ORINCON), Carol Nelepovitz (ORINCON), Bill Payne (MIT/LL), Scot Seto (ORINCON), and Art Teranishi (ORINCON). The author also wish to acknowledge LT John Schierling for his work in initiating this effort and promulgating the vision. The first author's funding has been received via NUWC contract N66604-95-D-0221, Leonard Cohen, COR, and Jim Ionata, NUWC technical monitor.

REFERENCES

D.L. Hall and J. Llinas, “An Introduction to Multisensor Data Fusion,” *Proceedings of the IEEE,* Vol. 85, No. 1, Jan. 1997.

M.E. Liggins II, C.Y. Chong, I. Kadar, M.G. Alford, V. Vannicola, and S. Thomopoulos, “Distributed Fusion Architectures and Algorithms for Target Tracking,” *Proceedings of the IEEE,* Vol. 85, No. 1, Jan. 1997.

K. Richards, W. Collier, R. Matis, and H. Hershey, “Common Acoustic Sensor Data Exchange (CASDE) File Format (Revision 1.3),” NUWC Document, Report No. ETC:96:08-019B, July 1997.

H. Schulzrinne, A. Rao, and R. Lanphier, “Real Time Streaming Protocol (RTSP),” Internet Draft draft-ietf-mmusic-rtsp-09.txt, Internet Engineering Task Force, Feb. 1998.

H. Schulzrinne, S. Casner, R. Frederick, and V Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Internet Draft draft-avt-rtp-new-00.txt, Internet Engineering Task Force, Nov. 1997.